

# **2016 International Conference on Platform Technology and Service (PlatCon)**

**Proceedings**

**15-17 February 2016  
Jeju, Korea**



IEEE Catalog Number: CFP16F03-ART (Xplore)  
ISBN: 978-1-4673-8685-2 (Xplore)

IEEE Catalog Number: CFP16F03-CDR (CD)  
ISBN: 978-1-4673-8684-5 (CD)

# SW Visualization Framework for Safe Software

So Young Moon

SELab., Dept. of Computer and  
Information Communication  
Hongik University  
2639, Sejong-ro, Jochiwon-eup,  
Sejong, 30016, Korea  
msy@selab.hongik.ac.kr

Bo Kyung Park

SELab., Dept. of Computer and  
Information Communication  
Hongik University  
2639, Sejong-ro, Jochiwon-eup,  
Sejong, 30016, Korea  
park@selab.hongik.ac.kr

R. Young Chul Kim

Dept. of Computer and Information  
Communication  
Hongik University  
2639, Sejong-ro, Jochiwon-eup,  
Sejong, 30016, Korea  
bob@hongik.ac.kr

**Abstract**— Today, it has been increasing software in most applications of most industry fields. Likewise, diverse industries (such as cars, aviation, and medical equipment) not only have adopted software, SW-related safety issues but also have emerged. Despite the concerns over SW quality, the competitiveness of local SW quality has diminished. With Safe SW technology, we need to reduce the socio-economic damages resulting from SW defects, and to enhance the competitiveness in global market. This paper proposes a SW visualization framework to develop error-free and safe software.

**Keywords**—*Function Point; Software Visualization; Reverse Engineering*

## I. INTRODUCTION

Software plays pivotal roles in society to the extent that most national infrastructure and large-scale industries rely on SW-based operation control. According to the data published by the National IT Industry Promotion Agency (NIPA) in 2006, the importance of SW per each product is growing by approximately 14.25% over 2002 in each industry field. Moreover, global manufacturing companies are trying to transform into software players. State-of-the-art aircrafts, ships and cars are equipped with sophisticated software. According to the data from the Science and Technology Policy Institute in 2015[1], the importance of SW started to outweigh that of HW as of 2002. In 2008, more than half (52.4%) of the development cost in the automobile industry was used for software-related electronic control features in cars. Similarly, software expenses took up 53.7% of electronic appliance development, 51.5% of industrial automation, and 52.7% of communication industry. As the scale and complexity of SW increases, issues of reducing development cost as well as securing safety and quality control have been raised. Yet, the quality of local SW has been decreasing. For a sustainable growth of ICT industry, it is necessary to secure the global competitiveness of local SW industry. Likewise, for local SW industry to secure global competitiveness, it is urgent to ensure the competitiveness of SW quality. To minimize the socio-economic damages attributable to SW defects, it is necessary to increase the global competitiveness of ICT convergence products with safe software. Specifically, it is desperately needed to develop technology viable for safe SW development so that local software SMEs having difficulties in competing

with foreign companies in global and local markets can overcome their limits. Currently, tools that are widely used to support the development and management of safe SW are mostly dominated by overseas large enterprises including IBM. Despite their functional superiority, such tools are so expensive that most local companies excluding larger enterprises cannot afford those foreign tools. Given the small-to-medium structure of local SW industry, it is necessary to support the development and management of safe SW by securing and applying the technology of open-source-based tools in lieu of the expensive foreign tools. Existing tools are far from integrated structures but adopt separate functions which make it difficult to apply and use all open source tools without much analysis. To implement safe SW, to create value in this smart era, and to shorten the time to market, a method of verifying the existing safe SW and analysing risks as well as visualization technology are necessary. As SW products reflect accumulated knowledge and experience, they are characterized by unique concepts of ownership and transaction, increasing social value via reuse and share, and development by man. Due to such attributes inherent in SW, advanced developers' competency is 28 times higher than that of low-level ones [2]. To minimize the difference in quality attributed to the experience and capacity varying with individual SW developers, the technology to automate and visualize design/implementation is needed. Also, despite the importance of techniques to improve the quality of new SW development, it is a more feasible approach to build and reuse safe SW based on the existing software owned by companies.

For error-free software and reliable codes, it is necessary to develop transparent and safe software through software visualization. This paper describes a safe software framework designed to visualize the software development, test and operation for safety. Chapter 2 covers related works on safe software and software visualization. Chapter 3 elucidates a visualization framework for safe software. Chapter 4 presents the conclusion and suggestion for further studies.

## II. RELATED WORK

### A. Safe Software

According to the National Safety Management Master Plan of the Ministry of National Security and Public Administration

(2010-2014) [3], safety refers to “a state free of natural, human or artificial threats, or a state of full readiness for such risk factors”. Based on the definition, software for safety may be defined as the software used to prevent natural, human or artificial risk factors, or to keep the full readiness. IEEE Std. (1228-1994) [4] defines software safety as preventing any incidents on system attributable to SW errors by eliminating SW risk factors. That is, software safety means a state free of SW risk factors.

Table. 1. Definition of Safe SW and SW Safety

Definition I	Definition II
<ul style="list-style-type: none"> <li>- SW for safety</li> <li>- SW conducive to safety</li> </ul>	<ul style="list-style-type: none"> <li>- SW safety</li> <li>- Safe SW</li> </ul>
<ul style="list-style-type: none"> <li>-SW capable of safely protecting and monitoring people or system to sustain safe society</li> <li>- SW conducive to safety in connection with ICT</li> <li>- SW conducive to addressing traffic, crime and environment issues</li> </ul>	<ul style="list-style-type: none"> <li>- Meaning SW itself is safe</li> <li>- SW should be free of errors or defects</li> <li>- Existing safe SW mostly focuses on nuclear power and cars</li> </ul>

In this software-centered society where most of national infrastructure and large-scale industries are controlled by software, dependency on software increases in finance, auto, train, aviation, power, defense, healthcare and education.

Consequently, the scopes and scales of damages resulting from incidents and accidents associated with software have increased. Also, due to the paradigm shift in disaster and safety policies and the increasing disasters related to climate change, the importance of safe SW continues to grow. Fig. 1 shows local and overseas cases of accidents resulting from SW issues. As a case of SW errors leading to loss of life, the temporary

suspension of flights from landings and takeoffs in the southwestern part of the US in 2015 was attributed to the air traffic control system overloaded and halted in the process of calculating the altitude and velocity of U2 reconnaissance aircrafts in the LA Air Route Traffic Control Center.

B. SW Visualization

Hongik University SW Engineering Lab’s software visualization[5][8][9] may be fit for high-quality software development at IT venture startups, SMEs and even established entities that are typically constrained by a lack of personnel and financial resources [5]. Software visualization is a technique intended for the betterment of software quality control and maintenance by visualizing and documenting source codes and development processes. Visualization helps manage the quality of SW development by overcoming the invisibility of software, which is the most difficult aspect in SW development, and by putting the overall process of SW development into perspective. This paper concerns using the SW visualization to minimize work burdens and applicability by proposing a method of documenting diverse outputs loaded up inside the system in the process of development.

III. SW VISUALIZATION FRAMEWORK FOR SAFE SOFTWARE

Conventional software engineering techniques used to systematically develop SW in accordance with a process ensure a certain level of SW safety, and involve such models as CMMI, TMMi and SW process certification for assessing the maturity of corporate SW development. Yet, certification-based safe and quality SW development is not feasible for SMEs because of cost and time. The proposed safe SW engineering is a method of developing safe SW that can perform prevention, protection and prediction by considering





	Example of accident	Damage	Reason
<p><b>Damage of human life caused by SW errors</b></p>	<p>'09 Washington Metro collision accident</p> 	<ul style="list-style-type: none"> <li>• 9 people death</li> <li>• 70 people serious and slight injuries</li> <li>• 6 cars derailed</li> </ul>	<ul style="list-style-type: none"> <li>• <b>System error</b></li> </ul>
	<p>'14 American southwest takeoff and landing suspended</p> 	<ul style="list-style-type: none"> <li>• 212 flights delayed</li> <li>• 21 flights canceled</li> </ul>	<ul style="list-style-type: none"> <li>• <b>System stops due to overload</b></li> </ul>
<p><b>Ignore a warning of safe SW</b></p>	<p>'10 BP company petroleum ship fire</p> 	<ul style="list-style-type: none"> <li>• The world 6 ranking oil company British Petroleum (BP) oil leak and fire on the petroleum ship</li> </ul>	<ul style="list-style-type: none"> <li>• Advanced Cement Modeling SW (Safe SW) is automatically alarms needs of add the space between oil pipe.</li> <li>• BP <b>ignored warnings</b>, oil spill tracking technology absence</li> </ul>
<p><b>Safety Control SW incomplete and person's carelessness</b></p>	<p>'15 Yeongjong Bridge collision accident</p> 	<ul style="list-style-type: none"> <li>• 106 collision accidents</li> <li>• 2 people dead</li> <li>• 73 people serious and slight injuries</li> </ul>	<ul style="list-style-type: none"> <li>• Failure visibility due to heavy fog</li> <li>• <b>Safety fridity</b> up to 100km per hour</li> <li>• <b>No safety systems</b></li> </ul>

Fig. 1. Examples of accident

not only SW development process but also SW products and their operation. Moreover, this paper employs the visualization to realize safety unlike previous studies on SW engineering. Fig.2 shows the SW engineering applying the safe SW visualization technique, where the visualization is applied to SW product development, operation and process to realize 3P, or prevention, prediction/warning and protection.

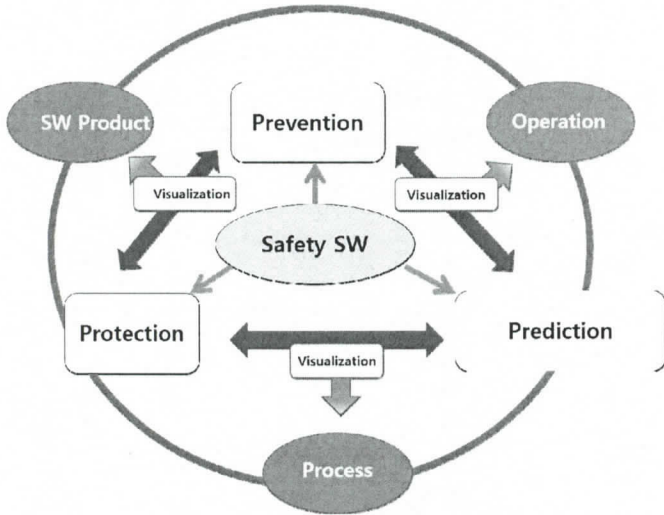


Fig. 2. Safety SW Visualization Approach

The present paper proposes a visualization framework enabling the implementation of safe software. Fig. 3 shows the safe SW visualization framework where SW visualization is used to develop reliable codes, error-free SW and transparent and safe SW.



Fig. 3. Safety SW Visualization Framework

### A. Hazard Analysis

A hazard refers to a state of potential risks (damage and failure) that may lead to an incident or accident [5]. Hazard analysis involves analyzing and assessing any hazards that might arise from unpredictable operation of devices or systems [6]. Hazard analysis is mentioned in ISO 26262[7] as one of the most important functional safety analysis activities.

### B. Risk Assessment

A risk refers to a combination of the likelihood of risky events or exposure and the severity of injuries or health hazards that might be caused by such events or exposure. Risk assessment is a process of assessing risks resulting from

hazards, and determining the tolerance against such risks by considering the fitness of existing management approaches.

### C. Software invisibility overcome

This paper applies the visualization to SW process, development and operation to overcome the SW invisibility. The visibility established by the SW Engineering Lab at Hongik University rids software of invisibility for the benefit of maintenance and SW quality enhancement [8]. Also, the Tool-Chain Method [9] is an open-source static analysis tool for SW visibility.

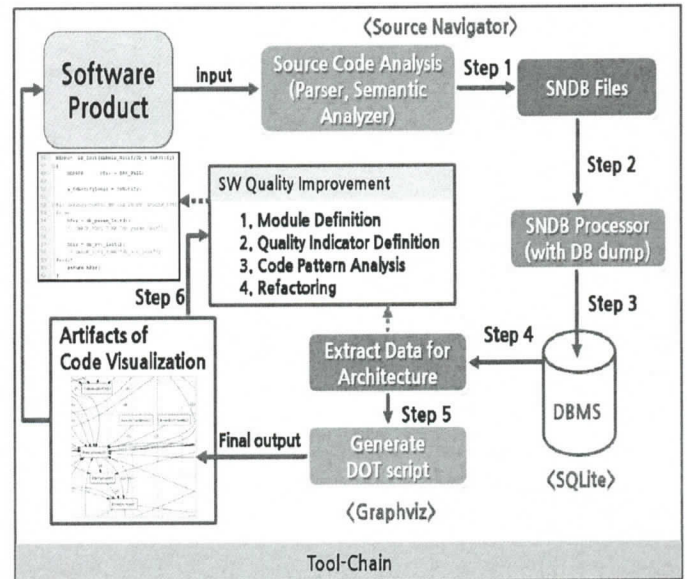


Fig. 4. Tool-Chain Method Process [9]

The Tool-Chain method comprises 6 steps: (1) Source Code Analysis, (2) SNDB Files Analysis, (3) Create DB from SNDB Processor, (4) Extract Data for Architecture, (5) Visualization, and (6) SW Quality Improvement. The SW modularization in (4) and (5) steps enables developers to visualize and reduce the complexity of codes for modification and rectification. Also, this method benefits SMEs or start-ups, which cannot afford costly CMMI, TMMi, and SW process certification for high quality SW products, by rectifying the quality and safety of software they develop. A company can use the process in Fig. 4 to improve the quality and safety of codes it develops. Fig. 5 shows the visualization of an Android app with the Tool-Chain Method Process. It shows 7 references to the SlidingTabStrip class in the SlidingTabLayout class, and 39 internal accesses of SlidingTabStrip. The SW visualization enables developers to visualize SW under development, to check the complexity of codes, and to review the relationship between classes regarding the use of methods and object generation. That is, visualization allows developers to proactively improve the quality of codes and thus to manage the quality constantly in the course of development. Also, in case a developer quits without leaving any development documentation, the visualization ensures the SW maintenance by generating outputs, e.g. the documentation on software design.

D. Safety SW Visualization Method

Efficient development and management of safe SW requires the visualization of safe SW development process. It is necessary to specify safety/quality goals, to engage in efficient safety/quality development activities, and to constantly monitor and control safety so as to overcome the SW invisibility. Also, companies should secure human resources capable of visualizing the SW safety development process and implementing the safety and quality of software for the benefit of corporate quality competitiveness. We suggest a tool supporting the SW safety visualization. Here, Subversion is used for the source code configuration management followed by Eclipse for development IDE, ASTM-based Parser for syntactic analysis, NSIQ Collector for measuring the complexity of codes, PMD and CPPCheck for static testing, Junit, CPPUnit and Unit for dynamic testing, TestLink for test case management, Redmine for request management, and Jenkins for integrated process management with a view to SW quality visualization. Furthermore, to secure safe SW, CAST, STPA, SpecTRM, OCRA, ILI and STPA-Sec are used for safety and risk analyses.

IV. CONCLUSION

Despite the efforts to support the development and management of safe SW, the inferiority of locally developed

tools and technology has resulted in the extensive dependence on overseas technology, which is far from a radical solution. Tools supporting the development and management of safe SW are so costly that most companies cannot afford such tools except a few large enterprises. Hence, for realizing safe SW, creating value in this smart era, and fulfilling the fast Time to Market, it is necessary to verify the existing safe SW, develop risk analysis methods, and adopt the visualization technology. The proposed framework for safe SW visualization is likely to reduce the scopes and scales of accidents due to SW errors. In addition, the framework is conducive to boosting the competitiveness of locally developed SW quality, given the rising quality control issues including development cost savings and safety on account of the increasing scale and complexity of software.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2013R1A1A2011601) and the Human Resource Training Program for Regional Innovation and Creativity through the Ministry of Education and National Research Foundation of Korea (NRF-2015H1C1A1035548).

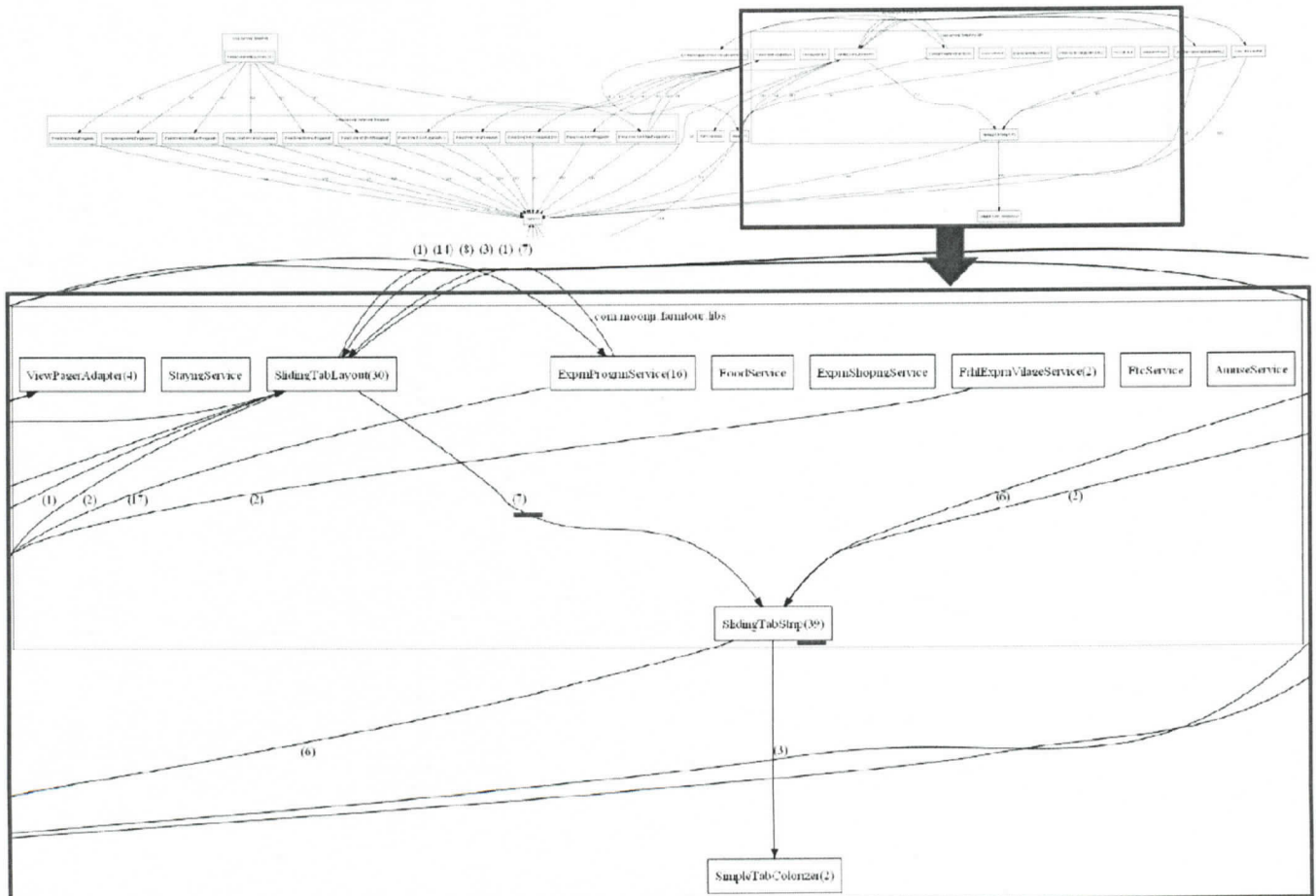


Fig. 5. SW Visualization about Android App [10]

## REFERENCES

- [1] Seung Hyun Kim, Man Jin Kim, "Analysis of Innovative Features in Software Use and Policy Direction", Science & Technology Policy Institute, May 2015.
- [2] Robert L. Glass, *Facts and Fallacies of Software Engineering*, Addison-Wesley, 2002. National Safety Management Master Plan, Ministry of Security and Public Administration(2010-2014)
- [3] IEEE STD 1228-1994. IEEE Standard for Software Safety Plans. 1994.
- [4] NASA TECHNICAL STANDRAD, NASA Software Safety Guidebook, 2004.
- [5] Geon-Hee Kang, R. Young Chul Kim, Geun Sang Yi, Young Soo Kim, Yong B. Park, Hyun Seung Son, 2015, "A practical Study on Code Static Analysis through Open Source based Tool Chains," *KIISE Transactions on Computing Practices*, vol. 21, no. 2, pp. 148-153.
- [6] Robyn R. Lutz, "Software Engineering for Safety: A Roadmap", ICSE '00 Proceedings of the Conference on The Future of Software Engineering, p.364-365, 2000.
- [7] ISO 26262-3:2011. Road vehicles—Functional safety. 2011
- [8] So Young Moon, Sang Eun Lee, R. Youngchul Kim, "Internal Code Visualization for Analyzing Code Complexity", *The 5th ICCT 2015.*, vol. 5, no. 1, pp. 268-269, June 2015.
- [9] So Young Moon, R. Youngchul Kim, "Code Structure Visualization with A Tool-Chain Method", unpublished.
- [10] Min-Gyu Park, Eun-Young Byun, Jeong-Wha Han, Robert Youngchul Kim, So-Young Moon, "Development of JDT Based Static Analyzer for Code Analysis", *The 2015 Fall Conference of the KIPS*, vol. 2, p.969-972, 2015.